# CS 1344 fall 98: Programming in C
# Project 3

Due November 20, 1998 at 11:59 PM

## Grade Report

Project 3 may be done jointly with one other class member. In that case, both names must appear in the program header as well as in the output (but as a single line). **For two of you to work together, you must turn in the signed collaboration form on November 3 in class**.

**Objective:**

This project will give you the opportunity to learn how to 1) declare and use arrays and **`struct`** variables in a C++ program to hold data temporarily, and 2) sort data held in a **`struct`** type array.

**Problem Statement:**

A professor maintains the following information on each student in a class: student ID number, status code (explained later), grades for five quizzes, grades for three assignments, grades for two tests, and grade for the final exam in a student file called "Project3.in". The grades are maintained in the following way. Each quiz score on the basis of 10 points, each assignment score on the basis of 100 points, each test score on the basis of 100 points, and the final exam score on the basis of 200 points. You are required to develop a C++ program to produce a grade report for the class based on the following processing requirements and input/output specifications:

The distribution of the weight for each item is as follows:

| | |
|---|---|
| Quizzes (five) | 10% |
| Assignments (three) | 35% |
| Test 1 | 15% |
| Test 2 | 15% |
| Final Exam | 25% |

According to the policy of the professor, for each valid or excused test absence, the percentage weight of the test (15%) is be added to the percentage weight of the final exam. In order to handle valid test absence cases, the professor maintains a data item ‘status code’ which is to be interpreted as follows for grading:

| Status Code | Meaning |
|---|---|
| 0 | Participated in all tests (or missed tests for no reasons) |
| 1 | Excused from Test1 |
| 2 | Excused from Test2 |
| 3 | Excused from both Test1 and Test2 |

Notice that status code is to be used to determine the correct percentage weight for the final exam, specifically for excused test absences.

Your program should calculate the total score for each student on the scale of 100, should round off the total score to one decimal place, and then should determine the letter grade for each student according to the following scale:

| Total Score | Grade |
|---|---|
| 90 - 100 | A |
| 80 - 89.9 | B |
| 70 - 79.9 | C |
| 60 - 69.9 | D |
| 0 - 59.9 | F |

**Input:**

To test your program, create an input file named "Project3.in" with a line of entry for each student in the class containing the following data items, each separated by one or more spaces.

Student ID (Maximum 5 digits)
Status code
Seven quiz scores each separated by space(s)
Six assignment scores separated by space(s)
Two test scores separated by space(s)
Final exam score

Assume all data are valid. Also assume that the file may contain entries for maximum one hundred students. You may use following data to test your program.

```
23028 1  7  8  5  6  7  48  58  84   0  50 166
10471 0  8  7  7  7  7  40  84  84  24  92 188
13302 3  8  4  6  8  0  70  67  71   0   0 198
 1824 1  7  0  4  6  7  45  70  92   0  34 157
12273 1  3  7  5  9  8  86  42  73   0  69 161
 1301 1  7  3  8  8  4  80  42  10   0  67 140
 1784 2  8  8  8  8  7  76  78  97  78   0 196
 6539 0  0  3  0  5  9  78  22  46  45  94 196
19283 1  8  0  7  9  8  33  67  91   0  59 180
10015 3  7  9  5  5  8  74  40  57   0   0 137
22816 3  9  6  5  8  8  70  52  81   0   0 192
 8612 2  8  8  8  5  9   0  30  69  50   0 168
28912 0  6  8  0  7  9  71  22  22  66  92 159
 1739 1  0  7  4  8  7  83  97  87   0  67 189
27266 2  3  9  5  7  7  31  64  92  88   0 122
31825 2  5  9  9  9  5  44  41  88  58   0 156
 1640 3  4  0  0  7  7  26  83  68   0   0 198
16557 1  3  8  4  8  7  93  73  78   0  20 178
20908 0  4  9  7  9  7  50  68  50  76  48 164
 9962 3  8  5  7  8  7  59  31  24   0   0 114
```

**Output:**

The program should create an output file 'Project3.out' to save the grade report for the class. With the sample input data, your program should generate the report in the form shown below. Notice that the report contains one blank line after the name(s) of the programmer(s), another after two heading lines and another blank line after the second dashed line near the end of the report. The student scores in the report table are sorted in descending order of overall scores of all students in the class. Do not forget to check your results by hand for each letter grade category. Note that the class average is to be computed after the grades (overall scores) have been rounded.

```
Name(s) of programmer(s): Jayan K. Nair

     CS 1000: Introduction to Computers
            Final Grade Report

 Student ID       Overall Score          Grade
-----------------------------------------------
    1784              88.0                B
    1739              84.2                B
   13302              83.9                B
   22816              83.7                B
    1640              78.7                C
   19283              73.5                C
   16557              73.1                C
   10471              72.4                C
   12273              72.4                C
   23028              69.5                D
   31825              67.5                D
   20908              65.9                D
    6539              65.8                D
   27266              65.6                D
    1824              65.5                D
   10015              64.4                D
   28912              63.0                D
    8612              60.3                D
    1301              59.5                F
    9962              51.7                F
-----------------------------------------------

Number of Students:    20
Class Average:       70.4
Number of A's:          0
Number of B's:          4
Number of C's:          5
Number of D's:          9
Number of F's:          2
```

**Hints:**

The following guidelines may help you to develop your solution program for the project:

- Use a struct type array in the program to hold data for students. Notice that the maximum class size is 100.
- Decompose the problem into many smaller sub-problems and then implement the solution of each sub-problem with a C++ function in the program.
- You may follow step-by-step process to develop your program. You may first write code for reading input data only. Once your are sure it reads data correctly from your test runs, you may write additional code (a function) to calculate overall scores for all students. Next when you are sure that your program does all calculations right for overall scores, you may add code (a function) to determine letter grades for all students. Once you make sure that it determines grades properly, you may add code to sort (a function) your data in decreasing order of overall scores to generate the final grade report. And the process continues until you have the complete program that produces the correct grade report.
- To reduce your burden, the code for sorting the student records is given below (You can use your own code for sorting if you wish).

```
// Here is the code for insertion sort which is efficient to sort a small number of
// Records. Insertion sort is similar to sorting a hand of cards by taking one
// card at a time from the table and putting it in its right place in the already sorted
// hand of cards.

// Student_Record is the type of the struct that is holding a single student data
// S_Data is an Array of size 100 holding 100 student records

void Sort(Student_Record S_Data[]/*in|out*/, int NumStudent/*in*/)     // Sort in descending order
{
        Student_Record temp;      // Create a temp space to store a single student record
        int i,j;                  // Loop variables

        for (i=1; i<NumStudent; i++)// Start from the beginning of the array
        {
                j=i;
                while ((j>0)&&(S_Data[j-1].score < S_Data[j].score))
                // Start from the new record which originally will be at the end of the already
                // sorted records (Similar to the new card that was taken from the table)
                {
                        temp = S_Data[j-1];       // When the new record's key value is higher
                        S_Data[j-1] = S_Data[j];  // than that of its previous record, move it
                        S_Data[j] = temp;         // backwards until it is in its correct
                        j--;                      // position
                }
        }
}
```

**Documentation:**

Your program should include all the documentation called for in the "Elements of Programming Style," including a header for the main program and readable in-line documentation that describes the purpose of each logical segment of code. White space and indentation for increased readability, a documentation header for each function you write, are also required. All function parameters must also be documented so that a user will understand how they are used.

**Submission:**

You should submit your source code (the *.cpp file) electronically to the automated grader. Do not submit the input or the output file. Before you submit your program, make sure your program runs perfectly and provides right output with the sample test data. Be aware that your program will be tested against a randomly generated input data. You will be allowed a maximum of four submits. Your submissions are archived by the grader and will be hand graded for the entire class for conformance to departmental documentation and coding style standards. This project should be submitted as Project Number 3.

**Distribution of Points:**

| | |
|---|---|
| Correct output | : 75% |
| Documentation | : 25% |
| Total | :100% |

**Bonus points and late penalty:**

| | |
|---|---|
| Bonus points | : 10 (2 per day up to a maximum of 10 points) |
| Late penalty | : 20% per each day late |